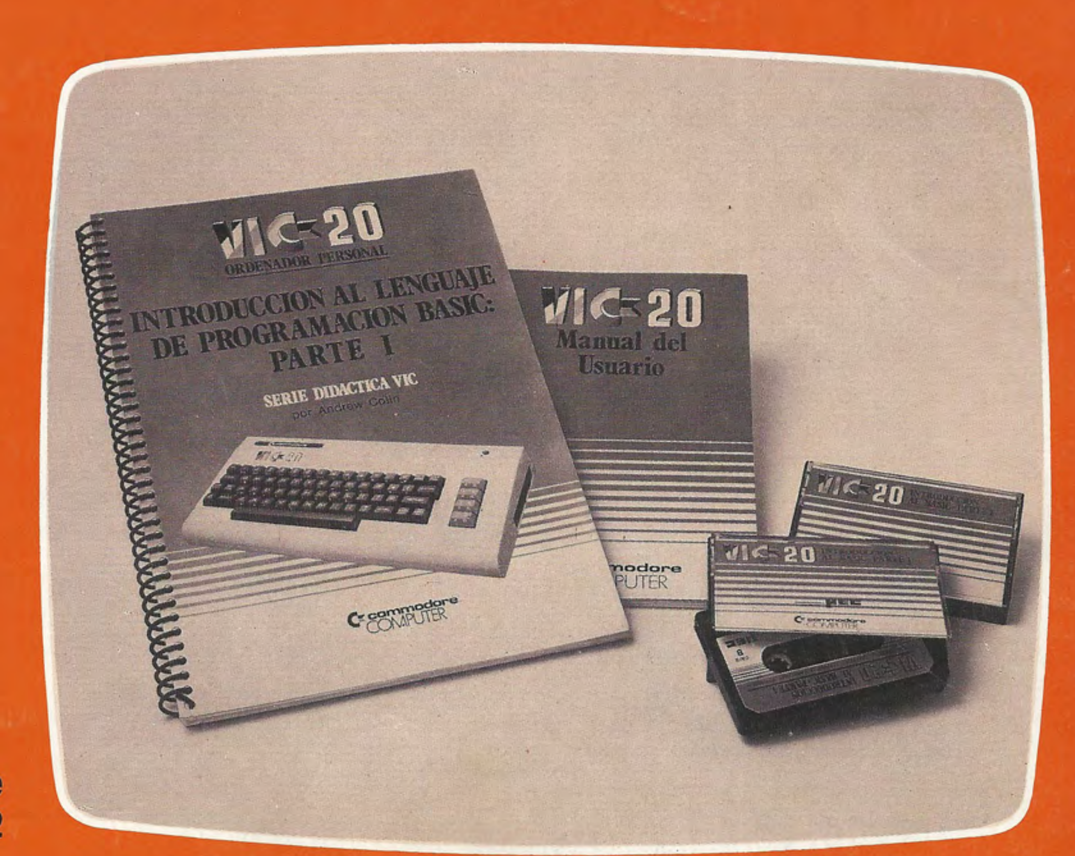
club commodore

Boletín informativo para los usuarios de microordenadores

VIC y CBM



Nº/2

noviembre 1982

I CONCURSO del "club commodore"

En este número 2 incluimos dos sencillos juegos para los más jóvenes (y también para los que se sienten en este feliz estado de espíritu). En el primero se simula el conocido juego del millón. Para mover la "pala" se deben pulsar las teclas M ("mover a la derecha") o Z ("mover a la izquierda"). En el segundo se trata de apostar sobre la posibilidad de que el número "secreto" esté o no entre los que da el ordenador. En caso de acertar, se suma a la cantidad actual el doble de lo apostado. En caso contrario, sólo se descuenta el monto de la apuesta (no todos los juegos de azar son tan benévolos).

Aprovechando estos programas vamos a convocar nuestro PRIMER CONCURSO, que va a consistir en lo siguiente: Estos programas pueden mejorarse (como TODOS los programas, para desesperación de los programadores) pero éstos son un poco más "mejorables" —quizá— que otros. Pues bien, las dos mejores mejoras (valga la redundancia) que recibamos antes del 31 de enero de 1983 no sólo serán publicadas en esta revista sino que sus autores recibirán como premio una cinta con todos los programas publicados en CLUB COMMODORE durante 1982. O sea: ¡a ver si nos animamos!

JUEGO DEL MILLÓN

```
10 REM MILLON
20 POKE36878,15
30 GOTO100
40 X=-X
50 POKE36876,200
60 RETURN
70 Y=-Y
80 POKE36876,220
90 RETURN
100 PRINT"■"
110 SC=0:Z=9
120 FORJ=23TO108STEP3
130 POKE7680+J,102
140 POKE38400+J,(INT(RND(1)*8))
150 NEXT
160 F=10
170 X=-3:Y=1
180 A=20:M=A
190 B=1:N=B
200 IFA<0THENGOSUB40
210 IFA>20THENGOSUB2000
220 A=A+X
230 IFB<1ORB>20THENGOSUB70
240 B=B+Y
250 IFPEEK(7680+22*A+B)=102THENGOSUB5000
260 POKE8163+F,32
270 POKE8165+F,32
280 POKE8164+F,102
290 POKE38884+F,0
300 GETA$
310 IFA$="Z"THENF=F-1
320 IFA$="M"THENF=F+1
330 POKE7680+22*M+N,32
340 POKE7680+22*A+B,42
350 POKE38400+22*A+B,0
360 POKE36876,0
370 M=A:N=B
380 GOTO200
2000 GOSUB40
2010 IFABS(F-B)<2THENRETURN
2020 Z=Z-1
2030 PRINT"■■BOLA"Z
2040 FORT=1TO20
2050 POKE36876,128+4*T:POKE36876,0
2060 NEXT
2070 IFZ=0THEN6000
2080 RETURN
5000 POKE7680+22*A+B,32
5010 POKE36876,0
5020 FORK=1TO10
5030 POKE36877,255-10*K:POKE36877,0
5040 NEXT
5050 SC=SC+367
5060 PRINT"■■BOLA"Z"■ TANTEO"SC
5070 GOSUB40
5080 RETURN
6000 POKE36876,0
6010 PRINT"■■■ FIN"
6020 GOTO6020

READY.
```

NOTA: Para concursar es necesario enviar a esta Redacción el programa grabado en cinta y, a poder ser, adjuntando un listado, o, en todo caso, una serie de explicaciones lo más claras posible utilizando la instrucción REM. Las cintas se devolverán en la primera ocasión al enviar la Revista pero —repetimos— es imprescindible remitir el programa en cinta (también se admiten discos).

APUESTAS

```
10 REM APUESTAS
20 D=20
30 A=INT(RND(1)*13)+1
40 B=INT(RND(1)*13)+1
50 IFABS(B-A)<2THEN30
60 C=INT(RND(1)*13)+1
70 IFA=CORB=CTHEN60
80 PRINT
90 PRINT"■■VD. TIENE ■";D;"PTS."
100 PRINT
110 PRINT"CUANTO APUESTA A QUE"
120 PRINT"MI PROXIMO NUM. ESTA"
130 PRINT" ENTRE";A;"Y";B;
140 INPUTE
150 IFE>DTHEN90
160 IFE<1THENPRINT"■C■O■V■A■R■D■E■!!"
170 FORZ=1TO1000:NEXTZ
180 PRINT"■MI NUMERO ES";C
190 IFE<1THEN280
200 IFNOT(C>AANDC<BORC<AANDC>B)THEN250
210 PRINT"BIEN HECHO"
220 PRINT"■VD. GANA";2*E;"PTS."
230 D=D+2*E
240 GOTO280
250 PRINT"■■LO SIENTO, VD. PIERDE:":PRIN
TE;"PTS.■"
260 D=D-E
270 IFD<1THEN370
280 FORZ=1TO2000:NEXTZ
290 FORZ=1TO24:PRINT
300 L=INT(RND(1)*500)+1
310 C=INT(RND(1)*8)+1
320 POKE7680+L,160
330 POKE38400+L,C
340 FORX=1TO50:NEXTX
350 NEXTZ
360 GOTO30
370 PRINT"EL JUEGO HA ■■TERMINADO!■"
380 PRINT"■ESTA VD. SIN BLANCA!!■"
390 FORZ=1TO1000:NEXTZ
400 GOTO380

READY.
```

GANADORES DEL SORTEO DE UN VIC-20, EN SONIMAG



En la foto puede verse a los ganadores del sorteo de un VIC realizado en Sonimag por Microelectrónica y Control S.A. De izquierda a derecha D. Félix Vicente Alcuaz, D. Juan Agustín Sánchez, Director de Microelectrónica y Control S.A. haciendo entrega a los afortunados de los equipos que les han correspondido y D. Ramón Querol i Batalla, el segundo nuevo propietario de un VIC-20. ¡Bienvenidos al grupo de usuarios!

Depósito Legal Sep. B. 2133-1958 - Altés, s.l., Caballero 87, Barcelona

EDITORIAL

creación del "Fondo de Programas de Ordenadores Personales Commodore"



Para que los lectores no digan que somos unos "pesados", en el número 1 no hicimos ninguna alusión al tema de las colaboraciones. Para estimular aún más a nuestros futuros colaboradores, vamos a dar ciertas facilidades e incentivos.

Por la presente se crea el "FONDO DE PROGRAMAS DE ORDENADORES PERSONALES COMMODORE" para intercambiar entre los que nos manden artículos o programas. El funcionamiento de este Fondo será como sigue: a todo aquel que nos mande un programa o un artículo con un mínimo de interés, se le remitirá, sin ningún cargo, una cinta de cassette con uno o más programas. En el momento en que el Fondo esté lo bastante crecido, el remitente podrá escoger el tema de su interés. Con esto pretendemos estimular el intercambio de programas y las colaboraciones en la revista.

VENTA Y COMPRA DE PROGRAMAS

Para aquellos de nuestros lectores que quieran sacarle "jugo" al resultado de sus peleas con BUG y crean que alguien puede comprar sus programas, les hacemos la siguiente oferta: En la sección "Marketclub" se insertará, para aquel que le interese, un anuncio en el que constará Nombre y Dirección, Teléfono, Título del programa, una —muy breve— descripción del mismo y el precio, para que cualquier potencial cliente pueda hacerse una idea de lo que se le ofrece y pueda ponerse en contacto con el autor. ¡Animo que muchas grandes empresas han empezado con menos!

AYUDA PARA LOS CLUBS DE USUARIOS

Para terminar hacemos un llamamiento a los usuarios para que se organicen en Clubs y nos comuniquen su existencia; tenemos una sección de esta revista reservada para recoger sus actividades y estamos dispuestos a prestarles toda la ayuda que precisen pero entendemos que deben ser los propios usuarios los que tienen que decidir cómo se organizan. ■

VENTANA CBM

grandes programas en poca memoria

por **JOAN CARLES SAMARANCH**

Cuántas veces hemos oído que la capacidad de tal o cual gran ordenador son 500 Kbytes ó 1 Mbyte (1.000.000 de caracteres!!), y hemos pensado que nosotros no podemos hacer programas demasiado complicados con nuestros «sólo» 32 Kbytes ó 16K ó 5K !! (en el caso del VIC-20).

Sin embargo, vemos programas para los equipos «Commodore» que manejan una cantidad de datos increíble. ¿Cómo es posible sin agotar la memoria?

«OVERLAYS»

Éste es el nombre técnico que se da al procedimiento de subdividir la aplicación en pequeños programas que se cargan entre sí. De esta manera, conseguimos, en cada momento, tener ocupada sólo una parte de la memoria aun cuando el total de la aplicación supere la capacidad máxima.

Existen dos modos de trabajo: guardando las variables — aunque cambiemos de programa — o sin guardarlas.

Utilizaremos los siguientes punteros de página cero: 42-43 final de programa-inicio variables, 52-53 final de memoria disponible y 201-202 final del último programa cargado desde cassette o disco.

LA INSTRUCCIÓN LOAD

Esta instrucción, que se utiliza para cargar programas en la memoria central, tiene dos peculiaridades si se utiliza dentro de programa: la primera es que hace un RUN automático después de ejecutarse y, la segunda, que sólo cargará parte del programa si la zona reservada para programas es menor que el propio programa. Cuando cargamos un programa se actualizan los punteros 201-202 y si el LOAD es directo también los 42-43.

Llegados a este punto, podemos probar los siguientes programas:

```
 5 REM PROGRAMA 1
10 POKE42, PEEK(201): POKE43,
   PEEK(202): CLR
20 FORI = 1TO10: PRINT
   «PROGRAMA 1»: NEXT
30 POKE42, PEEK(52): POKE43,
   PEEK(53): LOAD
   «PROGRAMA 2», 8
```

(pasa a la pág. siguiente)

grandes programas en poca memoria

(viene de la pág. anterior)

el segundo programa puede ser igual que éste pero cambiando los nombres. Si además le ponemos otra línea de programa podremos comprobar que es independiente de sus longitudes.

Lo realmente importante es que haya unas líneas como las 10 y 30 al principio y fin del programa, respectivamente. Estos «overlays» pueden funcionar indistintamente para «floppy» o para cassette, con la ventaja en disco, aparte de la rapidez, de que podemos hacer un bucle cerrado con los programas a ejecutar.

MANTENIENDO VARIABLES

El procedimiento para encadenar programas sin destruir las variables es el siguiente: de los programas de la aplicación se busca el más largo, es decir, el que, después de cargarlo, obtenemos el valor de PEEK(42) + PEEK(43)*256 más alto; anotamos estos valores, XX = PEEK(42) y YY = PEEK(43), y creamos la primera línea del primer programa de la aplicación tal que:

```
1 POKE42, XX: POKE43, YY:
  RUN10
10 REM PROGRAMA PARA ...
```

De esta manera, fijamos la configuración de memoria igual para todos los programas, aunque desaprovechemos espacio para la mayoría de ellos pero previendo el peor de los casos.

VERSIONES

Las posiciones de memoria mencionadas anteriormente son válidas para BASIC 4.0 y BASIC 2.0, respondiendo los demás a la tabla siguiente:

	BASIC 4.0/2.0	VIC-20	BASIC 1.0
Start of variables	42-43	45-46	124-125
Limit of memory	52-53	55-56	134-135
End of program	201-202	174-175	?

■

VIC-20

introducción al lenguaje de programación Basic. Parte I

Ante el número de llamadas recibidas preguntando por el Curso de Basic para el VIC-20 creemos que es interesante dedicarle unas líneas aunque sólo sea para desbloquear nuestra centralita.

La Introducción al Basic - Parte I se presenta en forma de libro acompañado de dos cintas de cassette con 17 programas, entre cuestionarios y ejemplos, todo ello traducido al castellano. El Curso, en su primera parte, se compone de 15 unidades didácticas y está estructurado como se puede ver en el índice de materias de tal modo que a cada unidad didáctica va asociado un programa (a veces dos) de los contenidos en los cassettes, siendo algunos de demostración o juegos pero la mayoría son programas pensados para que el usuario interaccione con el ordenador. Así, por ejemplo, el VIC le planteará una pregunta determinada y le presentará tres posibles soluciones. Si su respuesta fuese incorrecta, el VIC lo tendrá presente, le volverá a repetir la cuestión hasta que su respuesta sea la acertada pero, más adelante, volverá a plantearle la misma pregunta y verá si Ud. ha captado de manera definitiva la solución correcta.

Por otra parte no debe Ud. preocuparse del nivel de este curso pues parte de cero y de una manera clara y sencilla disecciona a los ojos del lector el lenguaje de programación BASIC.

ÍNDICE DE MATERIAS

Título	Temática	Programas grabados en el cassette
	Introducción.	
Unidad 1	**Preparativos:** Conexión del Sistema VIC; carga de programas desde el cassette; ajuste del receptor TV	CARTA DE AJUSTE AHORCADO
Unidad 2	**El teclado:** el cursor; signos gráficos, dibujo de figuras; escritura en pantalla.	VELMEC
Unidad 3	**Figuras en color:** control de los colores del fondo y marco de la pantalla; selección del color de los caracteres sobre fondo invertido.	UNI3CUEST
Unidad 4	**Comandos directos:** números y cadenas; el comando PRINT; efectos de la coma y punto y coma sobre el espaciado; variables; el comando LET; operadores aritméticos y de cadena.	UNI4EJER
Unidad 5	**Comandos almacenados** (en reserva); programas almacenados, comando GOTO; bucles simples (no controlados)	UNI5CUEST
Unidad 6	**Ayudas prácticas:** el comando LIST; edición de líneas; grabación y verificación de programas; errores frecuentes.	FRASES
Unidad 7	**Bucles controlados:** condiciones referentes a números y cadenas; control de bucle por contaje, etc.; el significado de «=» en BASIC.	UNI 7 CUEST
Unidad 8	**Identificación:** localización de errores	UNI8PROG
Unidad 9	**Colores programados:** modo de pantalla normal y especificado; representación en la pantalla de los caracteres de control; uso de los caracteres de control de posición y color en los programas; el reloj interno TI$.	UNI9CUEST
Unidad 10	**Entrada de datos:** el comando INPUT; relaciones entre el programador y el usuario.	UNI10CUEST
Unidad 11	**Diagramas de flujo:** comandos condiciones; validación de datos; diagramas de flujo; glosarios; diseño del programa.	UNI11PROG
Unidad 12	**Control avanzado de bucles:** los comandos FOR y NEXT; estructura de los programas.	UNI12CUEST
Unidad 13	**Sonidos:** las voces del VIC; control de altura, volumen y duración.	DEMOSOM PIANO
Unidad 14	**Programas de reducción de datos:** terminación de un flujo de datos; robustez de los programas.	CARA/CRUZ
Unidad 15	**Juegos:** tiempo de reacción; el comando GET: el cronometrador interno TI; la función RND; estructuración de juegos de azar.	REACCIÓN
	Recomendaciones finales.	
Apéndice A	Aspectos matemáticos del VIC: Expresiones - Precisión de cálculo - Funciones **normalizadas (standard).**	
Apéndice B	Solución de problemas (Respuesta a problemas)	
Apéndice C	Errores frecuentes	
Indice Alfabético		

RADIOAFICIÓN

utilización del VIC-20 como teclado de morse

Debido al interés mostrado por algunos usuarios-radioaficionados del VIC-20 abrimos en este número, lo que esperamos pueda ser, una sección fija dedicada a los amantes de tan noble hobby, sirviendo de punto de intercambio entre todos nosotros.

El programa que presentamos permite utilizar como teclado de morse el VIC-20, aprovechando al máximo sus prestaciones: control del transmisor a través del de la vía de acceso (port) del usuario — con un sencillo interfaz —, monitor de audio del televisor. La ventaja principal de esta aplicación es la de ofrecer a nuestro corresponsal una «manipulación» clara y precisa siendo ideal para empezar a familiarizarse con el sonido de CW.

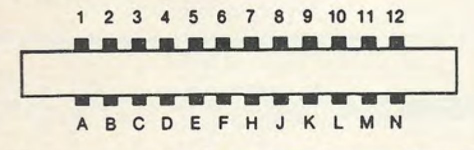
El programa incluye letras, números, el interrogante (?) y la barra (/); pudiendo además proveerle de un sencillo editor de mensajes preprogramados, de control de P.T.T. incorporado para trabajo en semi-«breaking», funcionamiento como baliza automática, etc. ¡¡Poniendo imaginación...!!

PROGRAMA

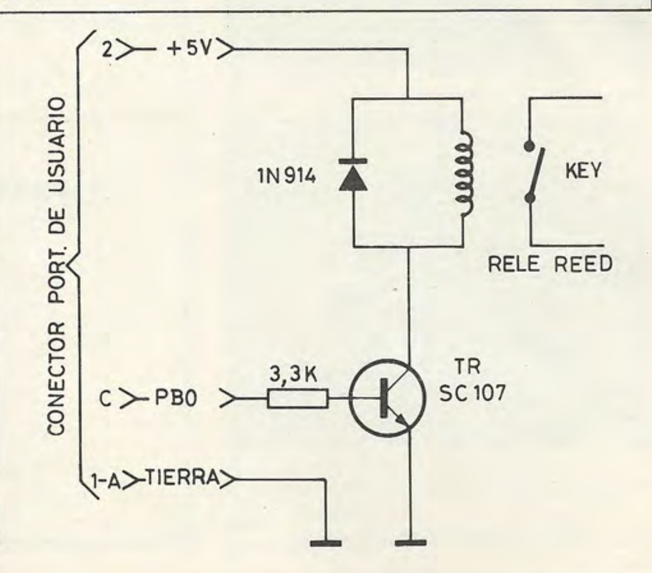
```
1 REM MORSE2.2
4 DIM M$(43)
5 POKE36876,241:POKE37138,1
15 PRINT"▢     EMISOR DE MORSE
16 PRINT"     ----------------    ▓▓▓"
20 GOSUB 300
25 PRINT"▓▓ENTRE MENSAJE ■"
30 GETS$:IFS$=""THEN30
35 PRINTS$;
40 GOSUB 500
50 GOTO30
100 END
300 REM SELECCION DE VELOCIDAD
310 PRINT"DURACION PUNTO    6"
320 PRINT"▢"TAB(15);:INPUTS
400 FOR I=0 TO 43:READ M$(I):NEXT I
405 RESTORE:DATA"-..-.","------"
410 DATA".----","..---","...--","....-"
420 DATA".....","-....","--...","---.."
422 DATA"----."
425 DATA"","","","","","..--..",""
430 DATA".-","-...","-.-.","-..","."
440 DATA"..-.","--.","....","..",".---"
450 DATA"-.-",".-..","--"."-.","---"
460 DATA".--.","--.-",".-.","...","-"
470 DATA"..-","...-",".--","-..-","-.--"
480 DATA"--.."
490 RETURN
500 REM
525 L$=CHR$(32):J=ASC(S$)-47:IFJ<0THENJ=
0:GOTO1000
530 IFJ>43THEN1000
540 L$=M$(J)
1000 REM SONIDO
1010 IF L$<>CHR$(32)THEN 1030
1020 GOSUB 7000:GOTO 1100
1030 W=LEN(L$)
1040 FOR I=1TO W
1050 X$=MID$(L$,I,1)
1060 IF X$=CHR$(46)THEN GOSUB 4000
1070 IF X$=CHR$(45)THEN GOSUB 5000
1080 NEXT I
1090 GOSUB 6000
1100 RETURN
4000 REM PUNTO
4010 FOR D=1 TO S:POKE36878,15:POKE37136
,1:NEXT D
4040 FOR D=1  TO S:POKE36878,0:POKE37136
,0:NEXT D
4070 RETURN
5000 REM RAYA
5010 FOR D=1TO 3*S:POKE 36878,15:POKE371
36,1:NEXT D
5040 FOR D=1 TO S:POKE36878,0:POKE37136,
0:NEXT D
5070 RETURN
6000 REM SEPARACION LETRA
6010 FOR D=1 TO2*S:POKE36878,0:POKE37136
,0:NEXT D
6040 RETURN
7000 REM ESPACIO
7010 FOR D=1 TO 6*S:POKE36878,0:POKE3713
6,0:NEXT D
7040 RETURN

READY.
```

ENTRADA/SALIDA USUARIO



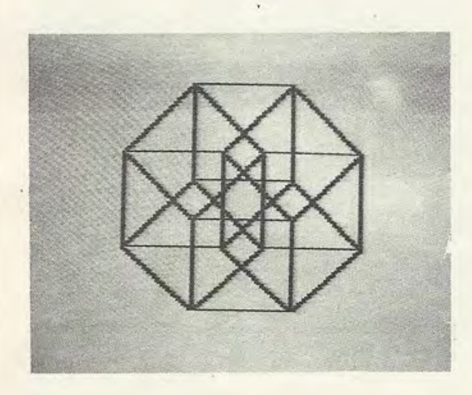
Contacto n.º	TIPO	NOTA	Contacto n.º	TIPO	NOTA
1	Tierra		A	TIERRA	
2	+ 5V	100mA MAX.	B	CB1	
3	RESET		C	PB0	
4	Joy 0		D	PB1	
5	Joy 1		E	PB2	
6	Joy 2		F	PB3	
7	Lápiz-óptico		H	PB4	
8	Interruptor cassette		J	PB5	
9	Entrada Serial ATN		k	PB6	
10	+ 9V	100mA MAX.	L	PB7	
11	Tierra		M	CB2	
12	Tierra		N	TIERRA	

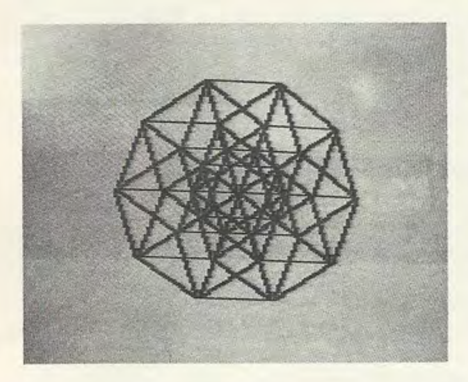


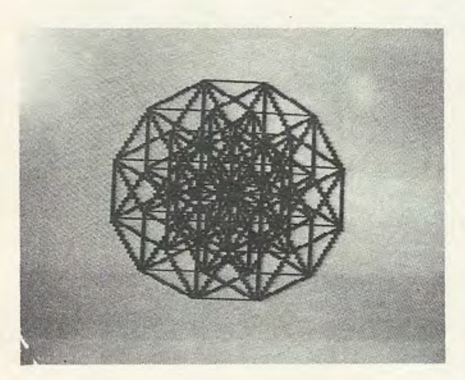
APLICACIÓN DEL SUPEREXPANDER

una ventana abierta a espacios de más de tres dimensiones

por **P. MASATS**







Teóricamente se puede describir un espacio de más de tres dimensiones (arriba-abajo, delante-detrás y derecha-izquierda) que es el que nosotros conocemos. De una manera muy primaria, podríamos explicar el aumento de dimensiones como sigue. Imaginemos un cubo. En un espacio de tres dimensiones — el nuestro —, es una figura que se caracteriza por tener una arista apuntando en cada dirección posible — en cada dimensión — de las tres que se unen para formar un vértice. El mismo caso se da en el cuadrado que es el equivalente bidimensional — de dos dimensiones — del cubo.

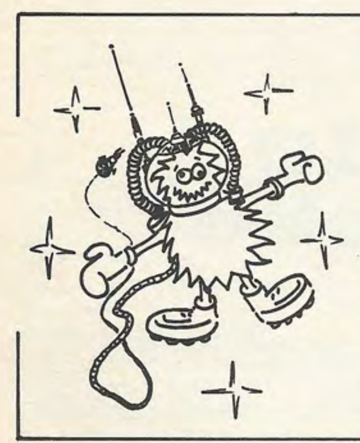
El problema se vuelve serio si se intenta imaginar la forma que tomaría un cubo en un espacio de cuatro o más dimensiones pues, para continuar siendo un cubo, debería tener una arista apuntando en la dirección de cada dimensión de este espacio, o sea, en el caso más sencillo de cuatro dimensiones, cada arista formaría un ángulo recto con cada una de las otras. Bien, esto es imposible visualizarlo incluso mentalmente dado que el único espacio en el que estamos acostumbrados a pensar es el tridimensional. Como mucho, podemos imaginar la «proyección» de una figura tridimensional en un espacio bidimensional — la sombra que proyecta una caja sobre un papel — o buscar algún truco para ver el aspecto de una figura multidimensional desde un espacio de un número de dimensiones que podamos manejar fácilmente.

Desde el punto de vista de nuestro ordenador personal este espacio es el de dos dimensiones (las de la pantalla) y el programa que está listado en la figura es el encargado de realizar los cálculos y el dibujo de lo que resultaría si pudiéramos proyectar cubos de diferentes dimensiones en la pantalla del VIC. Como ilustración se incluyen algunas fotografías de los resultados para diferentes números de dimensiones.

CORRESPONDENCIA

Para que las ofertas de colaboración para "Club Commodore", las peticiones de información, las consultas destinadas a la sección de "Correo abierto" o los textos para ser insertados en "Marketclub", lleguen prontamente a su destino, las cartas han de dirigirse a:

CLUB COMMODORE
"Microelectrónica y Control, S. A."
Taquígrafo Serra, 7, 5.ª planta
BARCELONA-29



PARA AQUELLOS LECTORES AFICIONADOS A LA CIENCIA-FICCIÓN, TRAEMOS ESTE PROGRAMA QUE LES PROPORCIONARÁ LA POSIBILIDAD DE "ECHAR UNA MIRADA" A UNAS ABSTRACCIONES MATEMÁTICAS QUE SE HAN CONVERTIDO EN UN LUGAR COMÚN DEL GÉNERO

PROGRAMA DE APLICACIÓN PARA EL SUPEREXPANDER

```
10 REM APLICACION PARA EL SUPEREXPANDER
20 REM P. MASATS; MICROELECTRONICA Y CON
TROL
30 REM 13 SEPTIEMBRE DE 1982
40 DIMX1(10),Y1(10),I1(10),I2(10)
50 GRAPHIC0
60 PRINT"J";
70 PRINT"**********************";
80 PRINT"*                    *";
90 PRINT"* PROYECCION DE CUBOS*";
100 PRINT"* DE N DIMENSIONES   *";
110 PRINT"*                    *";
120 PRINT"**********************"
130 INPUT"NO. DIMENSIONES";N
140 IFN>1THEN170
150 GRAPHIC0
160 END
170 IFN>10THEN130
180 P=π/N
190 I=-1
200 FORJ=1TONSTEP2
210 I=I+1
220 C=I*P
230 X1(J)=COS(C)
240 Y1(J)=SIN(C)
250 NEXTJ
260 I=N
270 FORJ=2TONSTEP2
280 I=I-1
290 C=I*P
300 X1(J)=COS(C)
310 Y1(J)=SIN(C)
320 NEXTJ
330 F=0
340 FORJ=1TON
350 F=F+Y1(J)
360 NEXTJ
370 X0=0
380 FORJ=1TON
390 IFX1(J)<0THENX0=X0+X1(J)
400 I1(J)=0
410 NEXTJ
420 F=190/F
430 X0=X0-40/F
440 Y0=0
450 OX=X0*F:OY=Y0*F
460 FORI=1TO2↑N-1
470 FORJ=1TON
480 I2(J)=I1(J)
490 NEXTJ
500 FORJ=1TON
510 IFI1(J)=1GOTO710
520 I2(J)=1
530 X=0
540 Y=0
550 FORK=1TON
560 X=X+I1(K)*X1(K)
570 Y=Y+I1(K)*Y1(K)
580 GRAPHIC2:COLOR1,1,0,0
590 NEXTK
600 GOSUB810
610 POINT2,XD,YD
620 X=0
630 Y=0
640 FORK=1TON
650 X=X+I2(K)*X1(K)
660 Y=Y+I2(K)*Y1(K)
670 NEXTK
680 GOSUB810
690 DRAW2TOXD,YD
700 I2(J)=0
710 NEXTJ
720 J=1
730 IFI1(J)=0GOTO770
740 I1(J)=0
750 J=J+1
760 GOTO730
770 I1(J)=1
780 NEXTI
790 GETA$:IFA$=""THEN790
800 GOTO50
810 GX=3:GY=4.7:OA=450:OB=80
820 XD=X*F*GX+OX+OA
830 YD=Y*F*GY+OY+OB
840 RETURN

READY.
```

MAPA DE MEMORIA DEL VIC-20 (III)

una ojeada al VIC propiamente dicho

(VIDEO INTERFACE CHIP)

por **PERE MASATS**

Aquí se presenta la continuación del artículo iniciado en la anterior edición con este mismo título. La Reg. 8 y 9 fueron los últimos a los que se hacía referencia. Por tanto, se continúa con los siguientes:

Reg. 10, 11 y 12. - 36874, 36875 y 36876 dec. - $900A, $900B y $900C

Estos registros manejan las tres voces de música del VIC, los valores que se almacenen en ellas deben ser mayores que 128 o la voz permanecerá en silencio. En cuanto a los valores de los POKE para determinadas notas ver el **Manual del usuario,** pág. 73 y, en general, todo el Capítulo 5 de dicho Manual.

Reg. 13. - 36877 dec. - $900D

Este registro es igual a los anteriores, excepto que, en vez de una nota, emite un ruido cuyo espectro variará con el valor que almacenemos en él. Si dicho valor es inferior a 128, esta voz permanecerá en silencio. El nivel de volumen de esta señal, junto con las generadas por los registros 10, 11 y 12, se controla mediante el contenido del registro 14.

Reg. 14. - 36878 dec. - $900E

Si la cantidad almacenada en este registro es menor que 16 se trata del valor del volumen de las voces del generador de sonido. Si es 16 o más, entra en juego otro factor: el multicolor.

Normalmente cada posición de carácter en el VIC contiene sólo dos colores: el del fondo y el del carácter. Si decidimos utilizar el modo multicolor podemos añadir dos colores al carácter: el del borde y otro que se puede seleccionar. Seleccionamos este color en la parte más significativa del registro 14. Si se divide el contenido de este registro por 16, y se descarta el resto, se obtendrá el valor del «color auxiliar».

De hecho, cada carácter de la pantalla del VIC se selecciona independientemente como «dos colores» o «multicolor» permitiendo una pantalla mezclada de los dos modos. Esto se consigue manipulando la tabla de colores.

Pruebe lo siguiente: borre la pantalla y escriba la letra «A» en la esquina superior derecha. Vaya a la línea siguiente y teclee POKE 38400,8. Ud. verá que, de repente, la letra «A» ha cambiado de forma misteriosamente, pero el resto de la pantalla permanece igual. Nótese que no hemos intervenido en el interfaz de video propiamente dicho sino en una dirección de memoria que está relacionada con una posición de pantalla. Para hacer las cosas correctamente debemos definir nuestro propio juego de caracteres.

Reg. 15. - 36879 dec. - $900F

Es el ultimo registro del vic pero uno de los más activos. Vamos a analizarlo en tres partes:

Divida el contenido de este registro por 16 y anote el resultado como «color de fondo de la pantalla». Ahora tome el resto, si es 8, o más, réstele 8 y anote: INV NO. El valor que queda entre 0 y 7 se anota como «color del cuadro».

El color del cuadro puede resultar muy útil. Constituye una señal muy fácil de realizar para llamar la atención del operador sobre una situación determinada sin afectar al contenido de la pantalla en sí mismo.

Si existe algún peligro, un error, o una explosión en un juego, puede transformarse el borde a rojo simplemente con POKE 36879,26 y volver al color normal con POKE 36879,27.

Otro ejemplo: en vez de dar un mensaje, como POR FAVOR ESPERE, mientras el programa realiza un cálculo largo, puede hacerle recorrer al borde una colección de colores que le asegurarán al operador (y lo que es más importante: al programador) que el programa no se ha parado.

EL COLOR DE FONDO DE LA PANTALLA Y SUS VARIACIONES

El color de fondo de la pantalla puede ser una ayuda sicológica muy interesante. Si se trabaja en un programa que deba recibir unos determinados tipos de datos, puede hacerse que, para cada uno de ellos, la pantalla tome un color característico ayudando así al operador a evitar confusiones. Pruebe POKE 36879,155 para ver el efecto.

Ahora vamos a ver esta misteriosa INV. Sabemos que podemos escribir caracteres de diferentes colores sobre un fondo de color único pero puede ser interesante hacer lo contrario: escribir caracteres del mismo color sobre fondos diferentes que pueden variar de carácter a carácter.

Pruebe POKE 36879,19. Borre la pantalla y escriba algunas letras, cambie el color y siga escribiendo... ¿Ve lo que ocurre? Está Ud. cambiando el color del fondo, no el del carácter.

Hay una gran cantidad de efectos que se pueden obtener «jugando» con el interfaz de video. Espero que, además de haber disipado sus dudas más acuciantes, este repaso a la estructura interna del componente más importante de su VIC-20, le haya servido de ayuda para mejorar sus programas. ■



COLABORACIONES

dos programas interesantes

por **JESÚS MELGAR VELASCO**

El programa de la figura 1 es un juego, todo consiste en evitar que caigan los ladrillos y, al acumularse por encima del nivel de la «pala» que utilizamos para pararlos, determinen el fin del juego. Dicha «pala» se mueve mediante las teclas «,» y «.».

El segundo es más didáctico — aunque no está nada claro el límite entre juego y aprendizaje — pues permite ver en la pantalla — ampliada — la estructura de los caracteres utilizados por el VIC. Debemos recordar que el conjunto de signos se reparten en dos grupos: los que se utilizan normalmente (MAYÚSCULAS y gráficos) y los que se obtienen al pulsar simultáneamente las teclas SHIFT y COMMODORE (MAYÚSCULAS y minúsculas). Como cada conjunto está formado por 256 signos tenemos un total de 512 (de 0 a 511). El conjunto de las minúsculas se obtiene entrando los números de 256 a 511. Ambos programas funcionan con cualquier capacidad de RAM.

El autor, en su carta nos dice que su nombre es Jesús Melgar Velasco, tiene 23 años, es un entusiasta de la informática y posee un VIC-20.

¡POR FIN¡ YA TENEMOS **COLABORACIONES.** J. MELGAR, UN USUARIO DE SALAMANCA DE 23 AÑOS DE EDAD, APROVECHANDO QUE SE SUSCRIBE A NUESTRA REVISTA —UNA EXCELENTE IDEA POR CIERTO— NOS ENVÍA DOS PROGRAMAS. ¡A VER SI TOMAMOS EJEMPLO!

```
1 POKE 36879,8:PRINT CHR$(5)
2 GOTO 22
3 CLR:C=0:Y=36878:Z=36877:POKE Z,255
4 W=3584:IF PEEK(648)<>16 THEN W=0
5 PRINT "♫ 0    LADRILLOS":P=8064-W::TI$=
"000000"
6 FOR X=8164-W TO 8185-W:POKE X,102:NEXT
7 K=PEEK(203)
8 IF K=29 AND P>8054-W THEN POKE P+1,32:
P=P-1
9 IF K=37 AND P<8074-W THEN POKE P,32:P=
P+1
10 POKE P,119:POKE P+1,119
11 B=B1:GOSUB 14:B1=B
12 B=B2:GOSUB 14:B2=B
13 GOTO 7
14 IF B<>0 THEN 17
15 B=INT(RND(TI)*22)+7702-W:POKE Y,15
16 FOR X=225 TO 255:POKE Z,X:NEXT:POKE Y
,0
17 IF PEEK(B+22)=160 AND B<8076-W THEN 3
3
18 IF PEEK(B+22)<>119 THEN 20
19 POKE B,42:POKE Y,15:FOR X=0 TO 100:NE
XT:POKE B,32:B=0:C=C+1:PRINT "█"C:RETURN
20 IF PEEK(B+22)=160 OR B>8141-W THEN B=
0:RETURN
21 POKE B,32:B=B+22:POKE B,160:RETURN
22 PRINT "♫██████████       PROGRAMA:",,,"
PARA-LADRILLOS",,,"             DE:"
23 PRINT "█       J.MELGAR"
24 FOR X=1 TO 4000:NEXT
25 PRINT CHR$(14) "♫    ¬♠—♠-L♠¯—\LLΓ●",
"                   "
26 PRINT "██¯EBES \\T¯\— QUE SE  TE AMO
NTONEN LOS...",,,"        L♠¯—\LLΓ●"
27 PRINT "█¯ARA CONSEGUIRLO...:",,,"
  ¬♠—♠LΓ●!!!"
28 PRINT "██¯ONTROL DE LA BARRA:",,," ´,
´......\ZQUIERDA."
29 PRINT "█ ´.´.........¯ERECHA."
30 PRINT "█ **¯ARA COMENZAR:",,," ¬/L●♠
//♠ |¯¯L♠."
31 GET G$:IF G$="" THEN 31
32 PRINT CHR$(142):GOTO 3
33 PRINT "████████  HAS SOBREVIVIDO..."
34 PRINT "█     "60*VAL(MID$(TI$,3,2))+VA
L(RIGHT$(TI$,2))TAB(9)"SEGUNDOS"
35 PRINT "█ PARA VOLVER A JUGAR",,," PUL
SA: ´S´=SI ´N´=NO"
36 GET G$
37 IF G$="S" THEN 25
38 IF G$="N" THEN SYS 58232
39 GOTO 36

READY.
```

Fig. 1

```
1 POKE 36879,8:PRINT CHR$(5):GOSUB 25
2 PRINT CHR$(142)"♫██  CHP$  DEL VIC - 2
0  ■":G=32768
3 PRINT"████████████████████ PARA TERMIN
AR...",,,"PULSE: ´ RETURN ´"
4 R$="":INPUT"█████████████████████CARACTER
 NRO.:           ██████████";R$
5 IF R$="" THEN SYS 58232
6 R=VAL(R$):P=R
7 IF P>255 THEN P=R-256
8 IF R>511 THEN 21
9 M$="MAYUSCULAS":IF R>255 THEN M$="MINU
SCULAS"
10 PRINT"████POKE X,                ██████
█████████████";P;M$
11 PRINT"███████             ────────────"
12 FOR L=0 TO 7:GOSUB 15
13 PRINT"     █████";PEEK(G+8*R+L);TAB(5);
"== ";L;TAB(11);" |";A$;"| ":NEXT
14 PRINT"              ────────────":GOTO 4
15 A=PEEK(G+8*R+L):A$=""
16 FOR B=1 TO 8
17 C=INT(A/2):D=A-C*2
18 IF D=0THEN A$=" "+A$
19 IF D=1THEN A$="●"+A$
20 A=C:NEXT:RETURN
21 PRINT"♫██  CHR$  DEL VIC - 20  ■"
22 PRINT"███████          █ ERROR ? ■"
23 PRINT"██████ *CANTIDAD ILEGAL!! ",,,"E
L NUMERO UTILIZADO ",,,"ESTA FUERA DE GA
MA."
24 FOR T=1 TO 4000:NEXT:GOTO 2
25 POKE 36865,170:PRINT"♫██████        PROGR
AMA :"
26 PRINT"███  CHR$ DEL VIC - 20"
27 PRINT"███   AUTOR  J.MELGAR"
28 PRINT"███   AGOSTO DE 1.982"
29 PRINT"███    MEMORIA UTIL :"
30 PRINT"███     1.5 K. BYTES"
31 FOR Z=170 TO 38 STEP-1
32 POKE 36865,Z
33 FOR T=1 TO 75:NEXT:NEXT
34 FOR T=1 TO 3000:NEXT
35 PRINT"♫█  —|—$  ¯¯L X\—   20  ■"
36 PRINTCHR$(14)" |E PUEDO MOSTRAR :",,,
"|ODOS LOS —|—$ QUE",,,"POSEO,COMPRENDID
OS"
37 PRINT"█ENTRE 0 Y 255, PERO",,,"PUEDES
 INTRODUCIR"
38 PRINT"█NUMEROS HASTA 511."
39 PRINT"█LOS NUMEROS MAYORES DE",,"255
CORRESPONDEN A MI",,,"FUNCIONAMIENTO EN
MODO"
40 PRINT"TEXTO (\INUSCULAS)."
41 FOR T=1 TO 12000:NEXT
42 FOR F=38 TO 170:POKE 36865,F
43 FOR T=1 TO 75:NEXT:NEXT
44 PRINT"♫":POKE 36865,38:RETURN

READY.
```

Fig. 2

CORREO ABIERTO

```
10 A=PEEK(203)
20 B=PEEK(653)
30 K=0
40 IFA=39THENK=1:GOTO90
50 IFA=47THENK=3:GOTO90
60 IFA=55THENK=5:GOTO90
70 IFA=63THENK=7:GOTO90
80 GOTO10
90 IFB>1THENB=0
100 K=K+B
110 PRINT"TECLA DE FUNCION"K"PULSADA"
120 GOTO10
READY.
```

Fig. 1

Pregunta: ¿Cómo puedo usar las teclas de función en un programa en Basic?

Respuesta: Con el programa de la figura 1 se tiene un control de la tecla de función pulsada. En la línea 110 tenemos el valor de la tecla como contenido de la variable K. Sustituyendo dicha línea por RETURN tenemos una subrutina que explora las teclas F1 a F8 hasta que se pulsa alguna de ellas.

Pregunta: ¿Cómo puedo seleccionar el «modo minúsculas» dentro de un programa?

Respuesta: Supongo que se refiere a pasar dentro de un programa y bajo control de éste del modo normal — MAYÚSCULAS y gráficos — al que permite utilizar el teclado como si se tratara de una máquina de escribir normal — minúsculas y MAYÚSCULAS —. Para ello el VIC-20 tiene dos juegos de caracteres, uno para cada modo, que se seleccionan al pulsar juntas las teclas SHIFT y COMMODORE. En modo programa esto puede lograrse cambiando con un POKE el contenido de la posición de memoria que «apunta» (por esto se le suele llamar «puntero») a la posición actual del generador de caracteres. Esta posición de memoria es la 36869 en decimal (9005 en Hex.) cuyos 4 bits menos significativos controlan la posición de esta tabla. El problema consiste en que los otros cuatro bits controlan la posición de la RAM de pantalla y un POKE demasiado alegre puede hacerla completamente ilegible. Además existe otro problema. Al ampliar la memoria a más de 8K, la dirección de la RAM de pantalla cambia, de manera que deberemos buscar un método que evite todos estos problemas. Cada uno de los dos juegos de caracteres ocupa en la ROM, llamada generador de caracteres, 2K de memoria. A su vez, los cuatro bits dan las direcciones en fracciones de 1K. Por lo tanto, si al contenido de este puntero le sumamos 2, éste nos apuntará 2K «más arriba» sin afectar para nada el resto del contenido del registro. (Para ampliar la información sobre este tema véase: CLUB COMMODORE NÚMERO 1; MAPA DE MEMORIA DEL VIC-20 (II), Pág. 4-6).

En definitiva para «pasar a minúsculas» teclee:

POKE 36869, PEEK (36869) + 2, y para volver al modo normal:

POKE 36869, PEEK (36869) — 2.

MARKETCLUB

En esta sección se dará buena acogida a los textos de ofertas o peticiones relativas a los diferentes modelos de micro-ordenadores "Commodore", a sus periféricos, a programas, a libros, a información... Extensión máxima por comunicación: cincuenta palabras.

- **VENDO Equipo CBM 3032 y 3042 (CPU y Floppy). Interesados llamar a Miguel al (93) 300 16 27 de Barcelona.**
- **DESEARÍA contactar con personas interesadas en la enseñanza de la Informática a jóvenes de 9-14 años para adquirir programas y documentación. Llamar a Eduardo Guardino al (93) 209 94 49 de Barcelona.**
- **VENDO Compilador de BASIC PETSPEED 2.0 para CBM 8032-8050, optimiza la velocidad un 50 %, prácticamente nuevo. Razón: José Luis Villalobos. Teléfono (93) 200 43 69.**
- **INTERCAMBIARÍA programas para el PET 2001 o CBM 3000. Ofertas: Santi EA3BVT. Teléfono 246 04 65. Apartado 5.350. Barcelona.**

Hex	Decimal	Description
0000-0002	0-2	USR jump
0003	3	Search character
0004	4	Scan-between-quotes flag
0005	5	Input buffer pointer; # of subscripts
0006	6	Default DIM flag
0007	7	Type: FF=string, 00=numeric
0008	8	Type: 80=integer, 00=floating point
0009	9	Flag: DATA scan; LIST quote; memory
000A	10	Subscript flag; FNX flag
000B	11	0=INPUT; $40=GET; $98=READ
000C	12	ATN sign/Comparison Evaluation flag
000D-000F	13-15	Disk status DS$ descriptor
0010	16	Current I/O device for prompt-suppress
0011-0012	17-18	Integer value (for SYS, GOTO etc)
0013-0015	19-21	Pointers for descriptor stack
0016-001E	22-30	Descriptor stack(temp strings)
001F-0022	31-34	Utility pointer area
0023-0027	35-39	Product area for multiplication
0028-0029	40-41	Pointer: Start-of-Basic
002A-002B	42-43	Pointer: Start-of-Variables
002C-002D	44-45	Pointer: Start-of-Arrays
002E-002F	46-47	Pointer: End-of-Arrays
0030-0031	48-49	Pointer: String-storage(moving down)
0032-0033	50-51	Utility string pointer
0034-0035	52-53	Pointer: Limit-of-memory
0036-0037	54-55	Current Basic line number
0038-0039	56-57	Previous Basic line number
003A-003B	58-59	Pointer: Basic statement for CONT
003C-003D	60-61	Current DATA line number
003E-003F	62-63	Current DATA address
0040-0041	64-65	Input vector
0042-0043	66-67	Current variable name
0044-0045	68-69	Current variable address
0046-0047	70-71	Variable pointer for FOR/NEXT
0048-0049	72-73	Y-save; op-save; Basic pointer save
004A	74	Comparison symbol accumulator
004B-0050	75-80	Misc work area, pointers, etc
0051-0053	81-83	Jump vector for functions
0054-005D	84-93	Misc numeric work area
005E	94	Accum#1: Exponent
005F-0062	95-98	Accum#1: Mantissa
0063	99	Accum#1: Sign
0064	100	Series evaluation constant pointer
0065	101	Accum#1 hi-order (overflow)
0066-006B	102-107	Accum#2: Exponent, etc.
006C	108	Sign comparison, Acc#1 vs #2
006D	106	Accum#1 lo-order (rounding)
006E-006F	110-111	Cassette buff len/Series pointer
0070-0087	112-135	CHRGET subroutine; get Basic char
0077-0078	119-120	Basic pointer (within subrtn)
0088-008C	136-140	Random number seed.
008D-008F	141-143	Jiffy clock for TI and TI$
0090-0091	144-145	Hardware interrupt vector
0092-0093	146-147	BRK interrupt vector
0094-0095	148-149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key down; 255=no key
0098	152	Shift key: 1 if depressed
0099-009A	153-154	Correction clock
009B	155	Keyswitch PIA: STOP and RVS flags
009C	156	Timing constant for tape
009D	157	Load=0, Verify=1
009E	158	Number of characters in keybd buffer
009F	159	Screen reverse flag
00A0	160	IEEE output; 255=character pending
00A1	161	End-of-line-for-input pointer
00A3-00A4	163-164	Cursor log (row, column)
00A5	165	IEEE output buffer
00A6	166	Key image
00A7	167	0=flash cursor
00A8	168	Cursor timing countdown
00A9	169	Character under cursor
00AA	170	Cursor in blink phase
00AB	171	EOT received from tape
00AC	172	Input from screen/from keyboard
00AD	173	X save
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00B1	177	Tape character parity
00B2	178	Byte received flag
00B3	179	Logical Address temporary save
00B4	180	Tape buffer character; MLM command
00B5	181	File name pointer; MLM flag, counter
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Tape writer countdown
00BB-00BC	187-188	Tape buffer pointers, #1 and #2
00BD	189	Write leader count; read pass1/2
00BE	190	Write new byte; read error flag
00BF	191	Write start bit; read bit seq error
00C0-00C1	192-193	Error log pointers, pass1/2
00C2	194	0=Scan/1-15=Count/$40=Load/$80=End
00C3	195	Write leader length; read checksum
00C4-00C5	196-197	Pointer to screen line
00C6	198	Position of cursor on above line
00C7-00C8	199-200	Utility pointer: tape, scroll

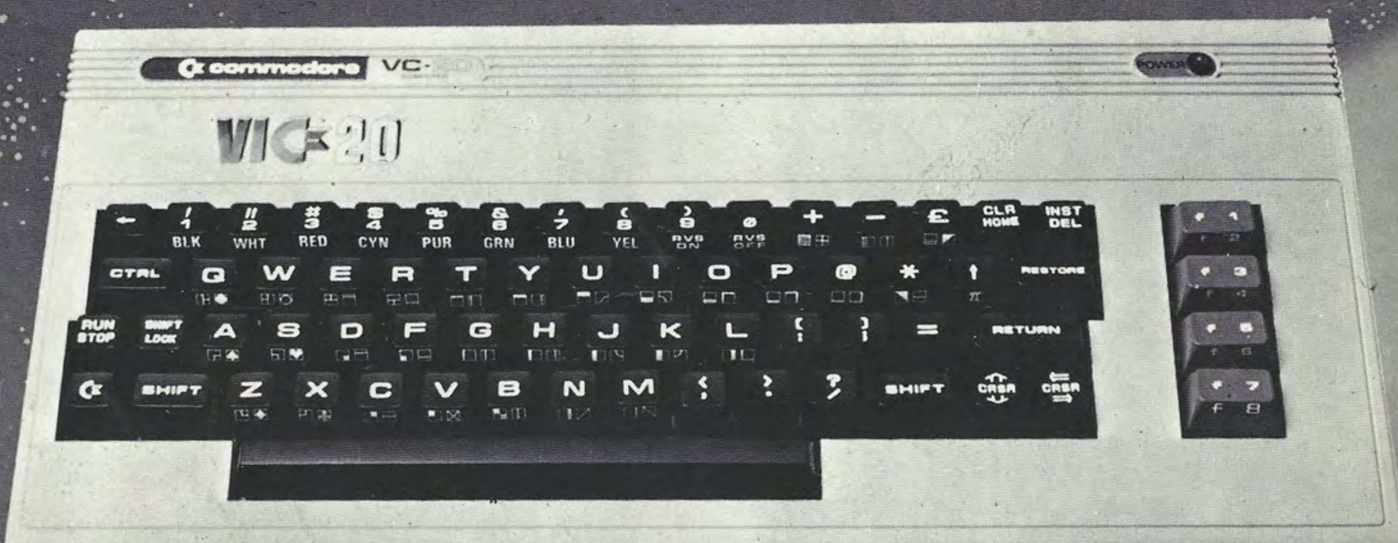
Hex	Decimal	Description
00C9-00CA	201-202	Tape end addrs/End of current program
00CB-00CC	203-204	Tape timing constants
00CD	205	0=direct cursor, else programmed
00CE	206	Tape read timer 1 enabled
00CF	207	EOT received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current file logical address
00D3	211	Current file secondary addrs
00D4	212	Current file device number
00D5	213	Right-hand window or line margin
00D6-00D7	214-215	Pointer: Start of tape buffer
00D8	216	Line where cursor lives
00D9	217	Last key/checksum/misc.
00DA-00DB	218-219	File name pointer
00DC	220	Number of INSERTs outstanding
00DD	221	Write shift word/read character in
00DE	222	Tape blocks remaining to write/read
00DF	223	Serial word buffer
00E0-00F8	224-248	(40-column) Screen line wrap table
00E0-00E1	224-225	(80-column) Top, bottom of window
00E2	226	(80-column) Left window margin
00E3	227	(80-column) Limit of keybd buffer
00E4	228	(80-column) Key repeat flag
00E5	229	(80-column) Repeat countdown
00E6	230	(80-column) New key marker
00E7	231	(80-column) Chime time
00E8	232	(80-column) HOME count
00E9-00EA	233-234	(80-column) Input vector
00EB-00EC	235-236	(80-column) Output vector
00F9-00FA	249-250	Cassette status, #1 and #2
00FB-00FC	251-252	MLM pointer/Tape start address
00FD-00FE	253-254	MLM, DOS pointer, misc.
0100-010A	256-266	STR$ work area, MLM work
0100-013E	256-318	Tape read error log
0100-01FF	256-511	Processor stack
0200-0250	512-592	MLM work area; Input buffer
0251-025A	593-602	File logical address table
025B-0264	603-612	File device number table
0265-026E	613-622	File secondary adds table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 input buffer
033A-03F9	826-1017	Tape#2 input buffer
033A	826	DOS character pointer
033B	827	DOS drive 1 flag
033C	828	DOS drive 2 flag
033D	829	DOS length/write flag
033E	830	DOS syntax flags
033F-0340	831-832	DOS disk ID
0341	833	DOS command string count
0342-0352	834-850	DOS file name buffer
0353-0380	851-896	DOS command string buffer
03EE-03F7	1006-1015	(80-column) Tab stop table
03FA-03FB	1018-1019	Monitor extension vector
03FC	1020	IEEE timeout defeat
0400-7FFF	1024-32767	Available RAM including expansion
8000-83FF	32768-33791	(40-column) Video RAM
8000-87FF	32768-34815	(80-column) Video RAM
9000-AFFF	36864-45055	Available ROM expansion area
B000-DFFF	45056-57343	Basic, DOS, Machine Lang Monitor
E000-E7FF	57344-59391	Screen, Keyboard, Interrupt programs
E810-E813	59408-59411	PIA 1 - Keyboard I/O
E820-E823	59424-59427	PIA 2 - IEEE-488 I/0
E840-E84F	59456-59471	VIA - I/O and timers
E880-E881	59520-59521	(80-column) CRT Controller
F000-FFFF	61440-65535	Reset, I/O handlers, Tape routines

Tal como anunciamos en nuestra edición anterior, seguimos publicando las notas técnicas en forma coleccionable. La finalidad de estas notas es la de constituir una referencia para complementar los artículos de la Revista y los manuales que acompañan a los equipos.

Aunque esta información se presenta en inglés, se espera que su utilidad no quede mermada ya que su comprensión quedará facilitada por las referencias que en los artículos se harán a estas notas. Por otra parte, esta reproducción directa en inglés asegura la plena fidelidad y evita cualquier error que pudiera originarse en caso de no mantenerse este procedimiento de transcripción.

VIC-20

EL ORDENADOR PERSONAL AMPLIABLE CON COLOR Y SONIDO.



49.500 Ptas.
COLOR-SONIDO

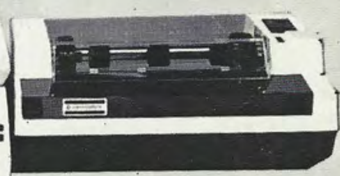
Así es el VIC-20

- Lenguaje BASIC extendido.
- Sistema operativo COMMODORE.
- 5 K RAM ampliable a 32 K.
- 16 colores, 4 generadores de sonido.
- 66 caracteres gráficos.
- Periféricos disponibles:
  - Cassette.
  - Impresora de agujas.
  - Unidad de disco de 170 K.

Así hace las cosas el VIC-20

- Enseña informática.
- Efectúa todo tipo de cálculos matemáticos.
- Realiza funciones docentes.
- Se encarga de múltiples tareas profesionales.
- Proporciona divertidos momentos de ocio.
- Ayuda a planificar labores domésticas.
- Hace todas las aplicaciones que Vd. imagine.





GRATIS

Con la adquisición de su VIC-20 recibirá además:
- MANUAL DEL USUARIO.
- INTRODUCCION AL LENGUAJE DE PROGRAMACION BASIC.
- Y 17 PROGRAMAS DE PRACTICAS (en dos cassettes).



commodore COMPUTER

Distribuidor exclusivo para España:

Microelectrónica y Control, S.A.
Taquígrafo Serra, 7 5.°. Barcelona-29
Princesa, 47 3.° G. Madrid-8

De venta en tiendas especializadas.

SLOGAN